
03.
One Dollar Book
WEB APPLICATION TECHNOLOGIES
By
Yas Sergersy

Web Application Technologies

Yasser Gersy

This book is for sale at
http://leanpub.com/webapplicationtechnologies

This version was published on 2016-10-18

Contents

Foreword

The best way to learn is by reading and practise , so here we are going to take you in a journey to discover web application technologies. If your are a web developer ,designer or a security researcher , this book will help you to simply understand web applications and how it works .

This book is written for those who are interested in web application and all its aspects , this will help you to reach your goal by posing a basic understanding of how these technologies work .

Intro

This book is written for anyone interested in studing web technologies , development , and web application hacking , no matter if you are a programmer or a hacker this book will be your guide to understand how web application works .

Thanks

I would like to thanks these people who helped and encouraged me to produce like this book : Hazem sabry , Peter yaworsk , Mohamed abdel baset , Rania Hassan and Karim robin.

Version

0.91

Chapter overView

1- Background

Explain What is HTTP and how it works , it's compnents cookies , urls , request response headers ..etc. ### 2- Functionality Explain the server side technologies. ### 3- oracle Java platform ### 4- Microsoft ASP.NET ### 5- PHP ### 6-Ruby On Rails ### 7- Structured query language ### 8- XML ### 9- Web Service ### 10- Client Side

Background

How Web Works?

Web is a bunch of connected systems , we are going to explain the HTTP, the most common protocol used nowadays . HTTP is standing for hyper text tranfer protocol, this protocol is used by these systems to communicate with each other , A system called ‘A’ sends a message to system B , and System B replies with another message to System A, with this simplicity we can call your browser (firefox , chrome ..etc) System a , when you write facebook.com on the navigation bar , you re communicating with system B which in this case will be Facebook , when facebook see your message it displays your facebook home page and you see your news feed , which is another message (response) sent by facebook to your browser.

![images/01-navbar.png]

Web applications employ a myriad of technologies to implement their functionalities they were developed for . We will examin the HTTP protocol , the technologoies commonly employed on both server and client sides .These technologies are in general easy to understand , this will help to easily build or attack a certain web application depends in your intention.

If you are familiar with web technologies , you can skim through this chapter and read another one , because this will dd nothing new for you.

if you still learning about web technologies , you should read this chapter first before reading later chapters , for further reading on many of the areas covered and advanced stage of understanding

HTTP and How web technologies work with HTTP?, i recommend HTTP World[1] it will cost you only 1.5$, you can request a free copy.

HTTP Protocol

Hyper text transfer protocol , is the core communications protocol and is used by all known web applications today.It is very simple protocol that was developed to transfer static-text resources and it hs since developed been extended nd leaveraged to allow it to enable building complex distributed application that we use everyday.

Http uses Message-Based module which a client sends a message and the server responds with another message .Http is essentially connectionless which means , each request and response use different connection although HTTP use the stateful TCP protocol as a transport mechanism so the request and response each one uses autonomus transaction and may use a different connection . 

HTTP Message

There are two types of message , when a client request a certain resource on a server , and the server replies with a responds , both request and responde are HTTP messages.

HTTP Request

Every HTTP message (Request and respond) Consists of Some headers , each on separate line Followed by blank line , followed by optional message body. An HTTP requests is as Follows:

[1] https://leanpub.com/HTTPWorld

```
1   GET /Topics/488/get.ashx?id=55&size=7 HTTP/1.1
2   Accept: application/x-ms-application, image/jpeg, app\
3  lication/xaml+xml,
4   image/gif, image/pjpeg, application/x-ms-xbap, applic\
5  ation/x-shockwave-
6   flash, */*
7   Referer: https://4neso.net/info
8   Accept-Language: en-us
9   User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Window\
10 s NT 6.1; WOW64;
11  Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.\
12 30729; .NET CLR
13  3.0.30729; .NET4.0C; InfoPath.3; .NET4.0E; FDM; .NET \
14 CLR 1.1.4322)
15  Accept-Encoding: gzip, deflate
16  Host: securitytraning.net
17  Connection: Keep-Alive
18  Cookie: SessionId=878FSA87SAADG4JH32323JHJ
```

The First line of HTTP Request consisit of Three parts

- A **verb** indicating the the method , the most common verb Are GET , is used to request a certain resource ,POST is used to send data or make an action on the server.
- A **url** , which is a string to derermine which resource is requested by the client , each url consist of a path to the requested resource and optional query string the previous request url is '/Topics/488/get.ashx?id=55&size=7' , a url is starting with a path to the resource , followed by question mark , followed by querystring , the query string is a key ,value pairs each one separated by & if many , this querystring contains some info that will be sent to the server or the application , the prevoius exapmle querystring is : id=55&size=7

- HTTP **version** , all http version are 1.0 and 1.1 , most web browsers use 1.1 by default.

Most Common Request Headers

Some of interesting points is Referer header: This header is sent by web browser to tell the server from where you came , an example : if you are reading an article about football on myblog.com and this article contains a url to a youtube video and you clicked on this url , your browser will request this page and set the referer header to myblog.com so when youtube see your request , it will know that you were reading an article on this blog.

User-Agent: This header is sent to provide some information about the client browser or other software that generated the request , note that most web browsers sends the mozilae firefox prefix for historical reasons , Web browser requires to tell the application it is compatible with mozila firefox , so it adds the prefix on the header of the request.

Host : This header is sent to determine which host name the client requests ,because the host is not sent in the first url , if you typed the following url in your browser : http://google.com/news your broswer will set the host header to google.com automatically , this header is neccessary when hosting multiple websites on the same server.

Cookie: This header contains additional parametrs that are requested by the server to destinguish between users and determine their settings or additional info.

HTTP Respone

HTTP Response is a message the same as The HTTP request , A typicall HTTP response is as follows: HTTP/1.1 200 OK Date: Sat,

17 Sep 2016 09:23:32 GMT Server: Microsoft-IIS/5.0 X-Powered-By: ASP.NET 4 Set-Cookie: UTrack=tI8rasfaf892Uu85SD9x Cache-Control: no-cache Pragma: no-cache Expires: Thu, 01 Jan 1970 00:00:00 GMT Content-Type: text/html; charset=utf-8 X-Frame-Options: Deny Content-Length: 940

<html><head><title> Welcome to Minya </title></head><body>Hello world </body> </html>…..

** HTTP response first line contains three parts: **

- First the **version** of HTTP used.
- Second a **status code** indicating the result of the request. 200 is the most common value and it mean that the request was received and successfully handled , and the requested resource is being returned.
- Third is a **further describtion** of the result of the request , it can be set to any value , most web browsers does not use it for any purpose.

Most common HTTP resonse Headers

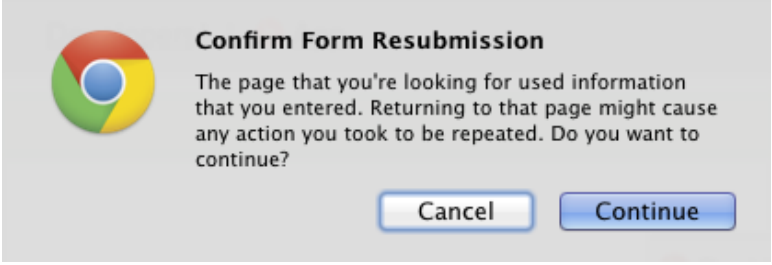
Date: indicating the date of the response created at. Server : Contains some information about the server technology and other software used and may contain some info about the server operating system, for security reasons this should not be sent in responses. X-Powered-By: Determine the web server technology used to handle requests , ASP.Net is a web language embeded with Microsoft IIS. Set-Cookie: A header asking the client browser to save cookie , for further requests , this cookie will be sent on subsequent requests. Cach-Control and Pragama : asking the client browser to store the response on it's cach or not. Expires: Determining the date of the response , this is neccessary for dynamic contents , when expires is set to expired date this asking the browser to delete the content and not store it. Content-Type: this header is used to indicate the type

of the body of the response message. X-Frame-options: is a header used for security reasons , which tell ask the browser not to display the response on iframes. Content-Length: This header indicates the size of the message body .

Every HTTP response contains an optional body following the blank line after the header .

HTTP Methods

When you are developing , attacking or just using web application , you will be dealling with the most common Methods Get and post , Get is used to request certain resource , While POST is used to send data or perform actions , Get does not contain a body , it can contains only parameters on url query string , while post can contain both , parametrs on query string , and those exist on the body , Urls are stored on bookmarks , browser history and server logs , so when urls are stored only parametrs on querystring are stored .Because the POST method is used to perform actions , for security reasons sensitive data should not be sent as url query string it should be exchanged using post body. when you click back on your browser to request the previous resource , if this resource requested using POST verb , your browser will warn you to prevent performing actions more than once.



In addition , HTTP support other methods that created for other purposes :

- **HEAD**: The same as GET except that the response should not contain a body, this method could be used to determine if a certain resource exists or not , before making a GET request for it.
- **TRACE**: This method is used for diagnostic purpose , The server should return the exact request sent by the client as a response body , this is helpful to detect the effect of any proxy exits between the client and the server.
- OPTIONS this method used to detect available methods for certain resouce , the server should send these methods on Allow header.
- PUT : is used to upload a resource to the server , for security reasons this method should not be allowed , as it may allow a malicious user to upload some scripts and executing it on the server.

There are many other methods , but not widely used so we are not going to discuss them.

URLS

Before explaining the urls , let's say that you are searching for certain article , you will look for it at google.com , then google redirects you to a blog , the blog tells you that you are searching with a wrong name , you go back to google and research again , you ended at another website , this website has a big archieve , you still searching inside it , finnally you found what you are looking for , need to save this to avoid this exausting process every time , so you will bookmark this page , you are not storing the page on your bookmarks , you are storing the actual url , this is URL with a simple expalaining . URLs are uniform resource locator for a web resource through wich this resource can be retrieved .

Url Format is as follows: protocol://[username:pass@]Host[:port]/[path]file[?querys

let's breack this expression , many parts are optional , The first word is the protocol used which will be HTTP or HTTPS , the username and password are optional credtentials depends on the system you are talking it not commonly used , the port used for the connection it exists only if it differs from the default , if not existed the client software uses the default port (80,443), the next is the path to the resource you are requesting , finally the query string which is optional contains some details for your request.

An example for a url is :

http://mysite.com/2015/9/articles.php?id=43

In addition , urls may be relative to host or certain path for example:

/2015/9/articles.php?id=43 or articles.php?id=43

This relatives are used only inside a web page , to allow navigation within the website.

URI uniform resouce indicator , this term is widely used on specification and it is just formal , just use url and forget about URI.

Representational state transfer REST is an style of architecture , that used by distributed systems in which requests and responses contain a representations of the current state of resource . A REST-Style url is this url which contains it's parametrs as file path and not a query string look at this example:

http://myegy.to/Search.php?k=ElJazeera&quality=HD

is corrosponding to the following rest-style url

http://myegy.to/search/Eljazeera/HD

HTTP Headers

HTTP supports a large number of Headers , there are some headers that could be used with the request and the response and others only used with one of these message types. This part will explain the most common headers you may encounter .

- *Connection*: is used to determine the other end of the communication to close the TCP connection after transmition is completed or wait for other messages.
- *Content-Encoding*: is used to determine the kind of the encoding of the message body , GZIP is most used , this allows to compress the body for faster transmission.
- *Content-Length*: specifies the length of the message body.
- *Transfer-Encoding*: specifies if any encoding is performed on the message body to facilitate transfer over HTTP.

General Headers

- *Ù*`  `*Accept* : determine the type of the content the client is willing to accept , such as Html files, images and documents.
- *Host* : indicates which host name , the client requesting.
- *Cookie* : contains the cookies , the server issued it before.
- *Accept-Encoding*: indicates which encoding the client willing to accept.
- *Authorization*: contains the client credetinals , if the server requested one of http authentication methods. -If-Modified-Since: sends the date when the browser last received the resouce , the resource changed the browser will receive it , if not changed since that time , the server tell the browser to use it's cached copy using a response with a status code 304. If-None-Match: contains an entity tag that is an identifier denoting to the message body , the browser submits this entity tag that the server issued with the requested resource to determine the client to use the cached copy or not.
- *Origin*: This is used for cross domain ajax requests , to indicate the domain from which the request originated, for seacurity reasons the browser only which can set the value of this header .
- *Referer* : this is used to indicate the url from which the request originated.

- *User-Agent*: contains some information about the client browser or other software that sent the request.

Response Headers

- *Access-Control-Allow-Origin*: Indicates if a request can be retrieved via cross domain ajax requests.
- *Cache-Control* : passing Cach directives to the client browser .
- *ETag* : this is an Entity tag , sent by the server , and browser can submit it back to the server in If-None-Match header , to inform the server which version of the resource the client caching.
- *Expires* : Tell the browser for how long this resouce is valid , browser can use the cached copy untill this time.
- *Location*: This header is used on redirection , to tell the browser the target of redirect , it is used with all response have status code starting with 3.
- *Pragma* : is thee same as cache-control , the only different between them is that cache-control is an HTTP 1.1 implementation while pragma is an HTTP 1.0 implementation , and some web browsers do not support cache-control , so this is the reason why pragma still in use.
- *Server* : provides information about the web server and other softwares used on the server side.
- X-Frame-Options: indicates whether the response body , is allowed to be embeded in a browser frame or not.

Cookies

Have you ever aksed your sefl , why a web application asks you only once for your username and password ? if you closed the current tab and reopened the page the application does not asks you again

, so how he knows this is you ?with you cookies. Cookies are a key value pairs , this like your passwords but it has an exapiration date , when a client sends a cookie to the server , it respond with sending other items or data ,These cookies are like other parametrs on the query string except that cookies are sent in every request your browser made .

The server asks the browser to store a cookie with this header : Set-Cookie: uid=34233532;

And your browser sent this cookie back besides old cookies if exist , using this header: Cookies: h=124124; size=255; lastshop=ab54x; uid=34233532;

Cookie are key value , the server can issue multiple cookies using multipe http resonse set-cookie headers , the browser can set the stored cookies in cookies header separated by a semicolon .

They can contain some attributes , which are set by the server , to control how the browser handle this cookie:

- Expires: is the date of expiration for this cookie , which will tell the browser to store this cookie untill the expire date is reached , then the browser will delete this cookie , if a server did not set the expires attribute , the broser will store the cookie untill the current session ends.
- Domaina : specifies which domain the cookie is valid for , each cookie should be valid for the domain sent from.
- Path : specifies a url which cookie is valid for.
- Secure : Asks the browser to send this cookie only via HTTPS.
- HTTPOnly: this attribute indicates that this cookie should not be accessed using javascript , to prevent session hijack via XSS .

Status Codes

Each Response mush contain a status code , to determine the result of the request , all status codes fall into five groups according to the code's first digit.

- 1XX : information, rarely to see this type of codes.
- 2XX : indicating that the requeste resouce existed .
- 3XX : used to redirect the client to a different resource.
- 4XX : used to notify the client , that it the request has some errors.
- 5XX : the server encountered an error while processing the request.

There are many status code for each groud , every code is used in specified circumstance.

- 100 Continue : is used when the request contains a body , and the server recieves it , the server sends a response with 100 status code to tell the client to send the rest of the body.
- 200 OK : the request is successful
- 201 Created : indicates that a file was created , used with PUT requests.
- 301 Moved permantely : Redirects the client permantely to a different resource.
- 302 Found : Redirects the client temporarily , this means that the resource exist but the client must be redirected to differnet resouce , then request the original resource .
- 304 Not Modified : Tell the browser to used the stored cached copy -400 Bad Request : used if a client sends a request contains an error , this happen when you send invalid method , or insert a space on host header or the url.
- 401 Unauthorized : Requires HTTP authentication.

- 403 Forbiddent : The client can not access this reource with out permission.
- 404 Not Found : The requested resource not found.
- 405 Method Not Allowed : This occures when you submit unsupported method , you may use DELETE while the server disabled it.
- 413 Request Entity Too Large: When you submit a long body , that the server can no handle.
- 414 Request URI Too Large: occures when the request url is too long.
- 500 Internal Server Error: This happen when you submit unexpected values .
- 503 Service unavailable : When the server can respond to the requested , but the application accessed by the server is crached or not responding.

HTTP Proxy HTTP Proxy is that mediates between client browser and server , if a browser or other client software is configured to use a proxy , the request will be sent to the proxy , and then the proxy pass it to the server , Server sends the respond back the proxy , and on the same way the proxy passes it to the client browser.Proxies also provide additional services like caching , authentication and access control .

HTTPS HTTP use TCP as transport mechanism, which is plain and unencrybted , so an attacker on the same network can intercept and view traffic , HTTPS is the same application layer protocol as HTTP , but tunneled over secure transport mechanism , secure sockets layer which protect the privace reducing the possiblities to interception attacks.

SSL Handshake Handshake is the process of exchanging data between the server and the client , to make sure the established connection is secured on these steps:

- Client sends SSL version number , random generated data , cipher settings and other information server needs to communicate with client.
- the server sends the SSL version number , random generated data , cipher settings and other information the client needs to communicate , and Server also sends it's own digital certificate.
- If the client requested a reource that requires authentication , the server request the client digital certificate.
- Client uses sent data and authenticate the server , if the server can not be authenticated , the client warns the user that encrybted connection could not be established.
- The client extracts the server public key from the server's digital certificate .All previous generated data is encrybted with the server public key to create pre master secret for session.
- If the server requested client authentication , the client signs new piece of data that is unique and known by both server and client , then send it along with encrybted pre master secret.
- if a client is requested to authenticate , the server attempts to authentticate the client , if the client can not be authenticated , the session is terminated , if client authenticated successfully the server uses it's private key to decrybt the pre master secret , then the server uses some steps as the client starting from pre master secret till generating the master secret.
- Both server and client use the master secret to generate keys to encrybt session keys to exchange data between them and verify the integrity .
- Client inform the server that all future message will be encrybted with session key.then sends a separated message indicating that the client portion of the handshake is finished.
- Server inform the client that all future messages will be encrybted with session key. then sends a separated message

indicating that the server portion of the handshake is done.
- SSL handshake is now completeted , and session is started , both server and client can exchange messages and use session keys to encrypt and decrypt these messages.

HTTP with proxy If a browser is configured to use a proxy , and a user requested a certain reource over unencrypted http connection , the browser places the full url in the request , and proxy server extract the path and the host and issue a request to the destination server.

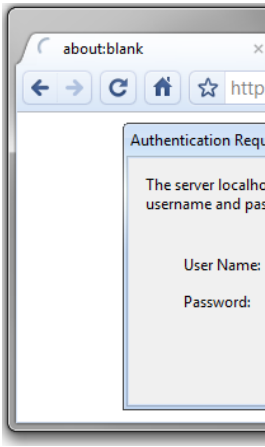
HTTTPS with proxy If a browser is configured to use a proxy server , the browser can not perform the ssl hand shake step with the proxy server , as it breacks the secure tunnel and leave the communications vulnerable to interception , so a proxy must act like a tcp-level relay.Hence the proxy sends the data in both directions between the server and the client. To allow the browser to perform the hand shake as normal.To enable this relay a browser sneds HTTP request with OPTION method to the proxy server indicating the destination hostname and port as url , if the destination is available , the proxy returns 200 ok , and keep the connection open , onward the proxy act like tcp-level relay to the destination server.

one of the amazing tools for hackers and web applications pentesters is Proxy , it allows them to stand between the server and the browser and alter both requests and responses exchanged between them.

HTTP Authentication

HTTP provide its own mechanismes for authentication using various schemes including the following:

- Basic : Browser asks the user for id and password , then base64 encode them and send the encoded string with each message to the server.



- NTLM : uses Windows NTLM authentication protocol.
- Digest: server send the client a string called nonce , the client uses MD5 Checksums of nonce and user creditentials.

These schemes are rarely to be used with most common web applications , it is used by organizations through theri intranet to access intranet-based services.

Functionalities

Web

In addition to HTTP server and client messages , Web application employ dozens of technologies that being accessed via web server.

Server-Side

in the past when web existed , all contents being delivered were static contents , like html and images , every time the server responds with the same content.Web applications are used to deliver dynamic content , every user request a resource the back-end application respond with different content. This content is being created on the fly. Dynamic contents is created using scripts or via executing codes on the server .When a client request a dynamic content , it does not ask the server to get a copy of the resource , the client may submit addional parameters , the server receive the request and pass it to the web application associated with the server , the application uses the data sent on the request to determine the contents that will be sent to back on the response to the client.

Parameters could be send as :

- URl query string
- Url path (REST-style)
- Cookies
- in request body.

In addittion , Server side may use any part to customize the contents, for example the server side may use user-agent to deliver certain content that is compatible with this browser.

Web applications are the same like computer programs , they employ numerous technologies , to deliver their functionalities.

- Web platform like ASP.NET and Java.
- Web scripting lanugaes like , PHP,Rails,VBscript.
- Servers like Apache ,IIS and nginx.
- Databases like MS-Sql and mysql.
- Backend components like directory services.